Package ‘gfoRmulaICE’

October 5, 2024

Type Package

Title ICE

Version 0.1.0

Description Implements iterative conditional expectation (ICE) estimators of the plug-in g-formula.
Both singly robust and doubly robust ICE estimators based on parametric models are available.
The package can be used to estimate survival curves under sustained treatment strategies (interventions) using longitudinal data with time-varying treatments, time-varying confounders, censoring, and competing events.
The interventions can be static or dynamic, and deterministic or stochastic (including threshold interventions). Both prespecified and user-defined interventions are available.

License GPL-3

Encoding UTF-8

LazyData true

Imports data.table,
ggplot2,
nnet,
doParallel,
parallel,
foreach,
stringr,
tidyverse,
dplyr,
rlang,
reshape2,
speedglm,
tuple

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.2

Depends R (>= 2.10)

Contents

`bootstrap_ice` *Bootstrap for ICE estimator*

Description

This function estimates the variance (and 95% confidence intervals) of the point estimates via non-parametric bootstrapping.

Usage

```
bootstrap_ice(
  f,
  K,
  nboot,
  coverage,
  parallel,
  ncores,
  ref_description,
  ref_intervention_varnames,
  total_effect,
  ref_intervention,
  interventions,
  intervention_varnames,
  intervention_description,
  intervention_times,
  ref_intervention_times,
  data,
  id,
  set_seed,
  ...
)
```

Arguments

`f`	a function specifying which ICE estimator to use for bootstrap.
`K`	a number indicating the total number of time points.
`nboot`	a number indicating the number of bootstrap samples.

`coverage` a number greater than 0 and less than 100 indicating the coverage of the confidence interval. Default is 95.

`parallel` a logical value indicating whether to parallelize the bootstrap process.

`ncores` a number indicating the number of CPU cores to be used in parallel computing.

`ref_description` a string describing the reference intervention in this bootstrap.

`ref_intervention_varnames` a list of strings specifying the treatment variables to be used for the reference intervention in this bootstrap.

`total_effect` a logical value indicating how the competing event is handled for the defined intervention. `TRUE` for total effect. `FALSE` for controlled direct effect.

`ref_intervention` a list of functions specifying the intervention to be used as reference.

`interventions` a list of functions defining the intervention to be used in this bootstrap.

`intervention_varnames` a list of strings specifying the treatment variables to be used for the defined intervention in this bootstrap.

`intervention_description` a string describing the defined intervention in this bootstrap.

`intervention_times` a list of numbers indicating the time points to which the defined intervention is applied.

`ref_intervention_times` a list of numbers indicating the time points to which the reference intervention is applied.

`data` a data frame containing the observed data in long format.

`id` a string indicating the ID variable name in `data`.

`set_seed` a number indicating the starting seed for bootstrap.

`...` any keyword arguments to be passed in `f`.

Value

A list containing the following components:

`ice_se` Standard error for the risk of the defined intervention.

`ref_se` Standard error for the risk of the reference intervention.

`rr_se` Standard error for the risk ratio.

`rd_se` Standard error for the risk difference.

`rr_cv_upper` The (100 - (100 - `coverage`) / 2)th percentile for the risk ratio at the last time point. Default is the 97.5th percentile when `coverage` is set to its default value 95.

`rd_cv_upper` The (100 - (100 - `coverage`) / 2)th percentile for the risk difference at the last time point. Default is the 97.5th percentile when `coverage` is set to its default value 95.

`ice_cv_all_upper` The (100 - (100 - `coverage`) / 2)th percentile for the risk of the defined intervention at all time points. Default is the 97.5th percentile when `coverage` is set to its default value 95.

ref_cv_all_upper
: The (100 - (100 - coverage) / 2)th percentile for the risk of the reference intervention at all time points. Default is the 97.5th percentile when coverage is set to its default value 95.

rr_cv_lower
: The ((100 - coverage) / 2)th percentile for the risk ratio at the last time point. Default is the 2.5th percentile when coverage is set to its default value 95.

rd_cv_lower
: The ((100 - coverage) / 2)th percentile for the risk difference at the last time point. Default is the 2.5th percentile when coverage is set to its default value 95.

ice_cv_all_lower
: The ((100 - coverage) / 2)th percentile for the risk of the defined intervention at all time points. Default is the 2.5th percentile when coverage is set to its default value 95.

ref_cv_all_lower
: The ((100 - coverage) / 2)th percentile for the risk of the reference intervention at all time points. Default is the 2.5th percentile when coverage is set to its default value 95.

ref_ipw_se
: Standard error for the observed risk.

ref_ipw_cv_all_upper
: The (100 - (100 - coverage) / 2)th percentile for the observed risk at all time points. Default is the 97.5th percentile when coverage is set to its default value 95.

ref_ipw_cv_all_lower
: The ((100 - coverage) / 2)th percentile for the observed risk at all time points. Default is the 2.5th percentile when coverage is set to its default value 95.

boot_data
: A list of data samples from all bootstrap replicates.

outcome_init
: A list, where each sublist contains the fitted outcome models in the first step of algorithm for the defined intervention from each bootstrap replicate.

comp_init
: A list, where each sublist contains the fitted competing models in the first step of algorithm for the defined intervention from each bootstrap replicate (if applicable).

np_model
: A list, where each sublist contains the fitted censoring and/or competing models in estimating the observed risk from each bootstrap replicate (if applicable).

outcome_by_step
: A list, where each sublist contains the fitted outcome models in each iteration of algorithm for the defined intervention from each bootstrap replicate.

comp_by_step
: A list, where each sublist contains the fitted competing models in each iteration of algorithm for the defined intervention from each bootstrap replicate (if applicable).

hazard_by_step
: A list, where each sublist contains the fitted hazard model, either time-specific models at each time point or one pooled-over-time global model, for the defined intervention from each bootstrap replicate (if applicable).

ref_outcome_init
: A list, where each sublist contains the fitted outcome models in the first step of algorithm for the reference intervention from each bootstrap replicate.

ref_comp_init
: A list, where each sublist contains the fitted competing models in the first step of algorithm for the reference intervention from each bootstrap replicate (if applicable).

`ref_outcome_by_step`
: A list, where each sublist contains the fitted outcome models in each iteration of algorithm for the reference intervention from each bootstrap replicate.

`ref_comp_by_step`
: A list, where each sublist contains the fitted competing models in each iteration of algorithm for the reference intervention from each bootstrap replicate (if applicable).

`ref_hazard_by_step`
: A list, where each sublist contains the fitted hazard model, either time-specific models at each time point or one pooled-over-time global model, for the reference intervention from each bootstrap replicate (if applicable).

`ref_data_err`
: A list of logical values indicating whether there is any data error for the reference intervention from each bootstrap replicate.

`ref_model_err`
: A list of logical valules indicating whether there is any model error produced from each bootstrap replicate for the reference intervention.

`ref_model_err_mssg`
: A list of strings for error messages of any model error from each bootstrap replicate for the reference intervention.

`ref_data_err_mssg`
: A list of strings for error messages of any data error from each bootstrap replicate for the reference intervention.

`compData` *Example Dataset for a Survival Outcome with Both Censoring and Competing Event*

Description

A dataset with 26581 observations on 10000 individuals and 4 time points. The dataset is in long format with each row representing the record of one individual at one time point.

Usage

```
compData
```

Format

A data frame with 26581 rows and 9 variables:

t0 Time index.

id Unique identifier for each individual.

L1 Binary covariate.

L2 Continuous covariate.

A1 Categorical treatment variable with levels 1, 2, and 3.

A2 Binary treatment variable.

C Censoring event indicator.

D Competing event indicator.

Y Outcome indicator.

compute_weighted_hazard   *Calculate Observed Natural Course Risk*

Description

This functions calculates the inverse probability weighted observed natural course risk.

Usage

```
compute_weighted_hazard(
  prob_censor,
  data,
  id,
  censor_varname,
  time_points,
  time_name,
  outcome_name,
  competing_varname,
  competing_fit,
  total_effect
)
```

Arguments

`prob_censor`	a vector of numerics specifying the estimated probability of censoring for each individual.
`data`	a data frame containing the observed data.
`id`	a character string indicating the ID variable name in `data`.
`censor_varname`	a character string indicating the censor variable name in `data`.
`time_points`	a numeric that indicates the total number of time points.
`time_name`	a character string indicating the time variable name in `data`.
`outcome_name`	a character string indicating the outcome variable name in `data`.
`competing_varname`	a character string indicating the competing event variable name in `data`.
`competing_fit`	a fitted model for the competing event model.
`total_effect`	a logical indicating whether to treat competing event as censoring or total effect.

Value

A list with the first entry as a vector of the mean observed risk. Its second entry is a vector of mean observed survival. Its third entry is a vector of inverse probability weight.

dynamic *Dynamic*

Description

This function specifies a dynamic intervention on the treatment variable specified in `data`. This function follows the treatment strategy specified in `strategy_before` until a user-defined condition that depends on covariate values is met. Upon the condition is met, the strategy specified in `strategy_after` is followed.

Usage

```
dynamic(
  condition,
  strategy_before,
  strategy_after,
  absorb = FALSE,
  id = id_var,
  time = time0var,
  data = interv_data
)
```

Arguments

`condition`	a string that specifies a logical expression, upon which is met, the strategy specified in `strategy_after` is followed.
`strategy_before`	a function or vector of intervened values that specifies the strategy followed after `condition` is met. The vector of intervened values should be the same length as the number of rows in the data frame `data`.
`strategy_after`	a function or vector of intervened values that specifies the strategy followed before `condition` is met. The vector of intervened values should be the same length as the number of rows in the data frame `data`.
`absorb`	a logical value indicating whether the strategy specified in `strategy_after` becomes absorbing (always treat with the specified strategy) upon the first time when `condition` is met.
`id`	a string specifying the ID variable name in `data`.
`time`	a string specifying the time variable name in `data`.
`data`	a data frame containing the observed data.

Value

a vector containing the intervened value of the same size as the number of rows in `data`.

Examples

```
data <- gfoRmulaICE::compData
# Dynamic intervention example 1: treat when L1 = 0, and not treat otherwise.
dynamic1 <- dynamic(condition = "L1 == 0", strategy_before = static(0), strategy_after = static(1),
absorb = FALSE, id = "id", time_name = "t0", data = data)
```

```
# Dynamic intervention example 2: never treat upon until L1 = 0, after which follows always treat.
dynamic2 <- dynamic(condition = "L1 == 0", strategy_before = static(0), strategy_after = static(1),
absorb = TRUE, id = "id", time_name = "t0", data = data)

# Dynamic intervention example 3: never treat upon until L1 = 0, after which follows natural course.
dynamic3 <- dynamic(condition = "L1 == 0", strategy_before = static(0), strategy_after = natural_course(),
absorb = FALSE, id = "id", time_name = "t0", data = data)
```

---

grace_period *Strategy with Grace Period*

Description

This function specifies an intervention in which treatment is initiated within the grace period of nperiod time units. During the grace period, the treatment variable follows its natural value or initiate intervention with a uniform distribution at each time point.

Usage

```
grace_period(
  type,
  nperiod,
  condition,
  data = interv_data,
  id = idvar,
  time_name = time0var,
  outcome_name = outcomevar
)
```

Arguments

type	a string specifying the type of grace period strategy. Possible values are "uniform" and "natural".
nperiod	a number indicating the length of grace period.
condition	a string specifying the logical expression, upon which is met, the treatment is initiated within nperiod time units.
data	a data frame containing the observed data.
id	a string specifying the ID variable name in data.
time_name	a string specifying the time variable name in data.
outcome_name	a string specifying the outcome variable name in data.

Value

a vector containing the intervened value of the same size as the number of rows in data.

Examples

```
data <- gfoRmulaICE::compData
grace_period <- grace_period(type = "uniform", nperiod = 2, condition = "L1 == 0", data = data,
                             id = "id", time_name = "t0", outcome_name = "Y")
```

ice *Iterative Conditional Expectation Estimator*

Description

This function implements iterative conditional expectation (ICE) estimators under user-defined treatment strategies. Available ICE estimators are classical and hazard-based pooling over treatment history ICE, classical and hazard-based stratifying on treatment history ICE, and doubly robust ICE estimators. See Wen et al. (2021) for more details regarding the parametric g-formula iterative conditional expectation estimator.

Usage

```
ice(
  data,
  time_points,
  id,
  time_name,
  outcome_name,
  censor_name = NULL,
  compevent_name = NULL,
  comp_effect = 0,
  outcome_model,
  censor_model = NULL,
  competing_model = NULL,
  hazard_model = NULL,
  global_hazard = F,
  ref_idx = 0,
  estimator,
  int_descript,
  ci_method = "percentile",
  nsamples = 0,
  seed = 1,
  coverage = 95,
  parallel = F,
  ncores = 2,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	a data frame containing the observed data in long format.
`time_points`	a number indicating the total number of time points.
`id`	a string indicating the ID variable name in `data`.
`time_name`	a string specifying the time variable name in `data`.
`outcome_name`	a string specifying the outcome variable name in `data`.
`censor_name`	a string specifying the censor variable name in `data`. Default is `NULL`.
`compevent_name`	a string specifying the competing variable name in `data`. Default is `NULL`.

comp_effect a number indicating how the competing event is handled for all the specified interventions. Default is 0. 0 for controlled direct effect. 1 for total effect.

outcome_model a formula specifying the model statement for the outcome.

censor_model a formula specifying the model statement for the censoring event. Default is NULL.

competing_model a formula specifying the model statement for the competing event. Default is NULL.

hazard_model a formula specifying the model statement for the hazard, if hazard-based estimator is used. Default is NULL. If specified, the model in hazard_model will be used. If NULL, the model in outcome_model will be used.

global_hazard a logical value indicating whether to use global pooled-over-time hazard model or time-specific hazard models, for hazard-based pooled ICE only. If TRUE, use pooled-over-time hazard model. If FALSE, use time-specific hazard models. Default is FALSE.

ref_idx a number indicating which intervention to be used as the reference to calculate the risk ratio and risk difference. Default is 0. 0 refers to the natural course as the reference intervention. Any other numbers refer to the corresponding intervention that users specify in the keyword arguments.

estimator a function specifying which ICE estimator to use for the estimation. Possible inputs are:

- Classical pooling over treatment history ICE estimator (classical pooled ICE): pool(hazard = F)
- Hazard-Based pooling over treatment history ICE estimator (hazard-based pooled ICE): pool(hazard = T)
- Classical stratifying on treatment history ICE estimator (classical stratified ICE): strat(hazard = F)
- Hazard-Based stratifying on treatment history ICE estimator (hazard-based stratified ICE): strat(hazard = T)
- Doubly robust weighted ICE estimator (doubly robust ICE): weight(treat_model) where treat_model is a list specifying the treatment model.

int_descript a vector of strings containing descriptions for each specified intervention.

ci_method a string specifying the method for calculating the confidence interval, if nsamples is larger than 0. Possible values are "percentile" and "normal." Default is "percentile."

nsamples a number larger than 0 indicating the number of bootstrap samples. Default is 0.

seed a number indicating the starting seed for bootstrapping. Default is 1.

coverage a number greater than 0 and less than 100 indicating the coverage of the confidence interval. Default is 95.

parallel a logical value indicating whether to parallelize the bootstrap process. Default is FALSE.

ncores a number indicating the number of CPU cores to use in parallel computing. Default is 2.

verbose a logical specifying whether progress of the algorithm is printed. Default is TRUE.

... keyword arguments to specify intervention inputs. If stratified ICE is used, keyword arguments also allow intervention-specific outcome models and competing models.

To specify interventions, please follow the input convention below:

- Each intervention is specified using the keyword argument name with *intervention* prefix.
- Use *i* after *intervention* prefix in keyword argument name to represent the ith strategy.
- Use . followed with *treatment variable name* after *interventioni* in keyword argument name to represent the treatment name within the ith strategy.

Each input of intervention keyword arguments is a list consisting of a vector of intervened values and an optional vector of time points on which the intervention is applied. If the intervention time points are not specified, the intervention is applied to all time points. For example, an input considers a simultaneous intervention with always treat on A1 and never treat on A2 at all time points looks like:

```
intervention1.A1 = list(static(1))
intervention1.A2 = list(static(0))
```

The above intervention applies to all time points. The following is an example of custom intervention time points, with always treat on A1 at time point 1 and 2 and never treat on A2 at time point 3 to 5.

```
intervention1.A1 = list(static(1), 1:2)
intervention1.A2 = list(static(0), 3:5)
```

If there is no intervention keyword argument specified, the function returns the natural course risk only. Please see the "Examples" section for more examples.

To specify different outcome model and/or competing model for different intervention, please follow the input convention below:

- Each outcome model is specified using keyword argument name starting with *outcomeModel* or *compModel* prefix for outcome model or competing model correspondingly.
- Use *.n* after *outcomeModel* or *compModel* prefix in keyword argument name to specify which intervention being applied to, where *n* represents the *n*th intervention.

The input to each outcome or competing model keyword argument is a model statement formula. If no outcome model and competing model keyword argument is specified, the models specified in `outcome_model` and `comp_model` are used. Please refer to the "Examples" section for more examples.

Details

Users could specify which version ICE estimator to use through `estimator`.

- `pool(hazard = F)` specifies the classical pooling over treatment history ICE estimator.
- `pool(hazard = T)` specifies the hazard-based pooling over treatment history ICE estimator.
- `strat(hazard = F)` specifies the classical stratifying on treatment history ICE estimator.
- `strat(hazard = T)` specifies the hazard-based stratifying on treatment history ICE estimator.

- `weight(treat_model)` specifies the doubly robust weighted ICE estimator where `treat_model` specifies the treatment model.

To provide flexible choices on model inputs for stratified ICE and doubly robust ICE, we allow users to specify intervention-specific model statements through keyword arguments. In the case where intervention-specific model statements are specified, treatment variables that are not intervened under some strategies will be considered as a covariate and automatically added into the model specification at each time point. Please see more details on how to specify intervention-specific model specifications in the "Arguments" section.

For example, the following input specifies `Y ~ L1` as outcome model for intervention 1, `D ~ L1` as competing model for intervention 2, `D ~ L1 + L2` as competing model for intervention 1, `Y ~ L1 + L2` as outcome model for intervention 2.

```
fit_hazard_strat <- ice(data = data, K = 4, id = "id", time_name = "t0",
outcome_name = "Y", censor_name = "C", competing_name = "D",
estimator = strat(hazard = T), comp_effect = 1,
censor_model = C ~ L1 + L2, ref_idx = 0,
int_descript = c("Static Intervention", "Dynamic Intervention"),
outcome_model = Y ~ L1 + L2,
competing_model = D ~ L1 + L2,
intervention1.A1 = list(static(0)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic("L1 > 0", static(0), static(1), absorb = F)),
intervention2.A2 = list(dynamic("L2 == 0", static(0), static(1), absorb = T)),
outcomeModel.1 = Y ~ L1,
compModel.2 = D ~ L1
```

Because the keyword argument for outcome model is not specified for intervention 2, the outcome model for intervention 2 is `Y ~ L1 + L2` as specified in `outcome_model`. Similarly, because the keyword argument for competing model is not specified for intervention 1, the competing model for intervention 1 is `D ~ L1 + L2` as specified in `competing_model`. In the case of controlled direct effect, the keyword arguments for competing models are ignored. Please see more examples in the "Examples" section.

Both built-in interventions and user-defined interventions are available.

The following are the built-in intervention functions in the package:

- Static Intervention: `static(value)` specifies a constant intervention with `value`.
- Dynamic Intervention:
  `dynamic(condition, strategy_before, strategy_after, absorb)` specifies a dynamic intervention where the strategy in `strategy_before` is followed until `condition` is met. Upon `condition` is met, the strategy in `strategy_after` is followed. If absorb is `TRUE`, the intervention becomes absorbing once `condition` is met.
- Threshold Intervention: `threshold(lower_bound, upper_bound)` specifies a threshold intervention. If the treatment value is between `lower_bound` and `upper_bound` inclusively, follow the natural value of the treatment. Otherwise, set to `lower_bound` or `upper_bound`, if the treatment value is below `lower_bound` or above `upper_bound`, correspondingly.
- Grace Period: `grace_period(type, nperiod, condition)` specifies a dynamic intervention with grace period. Once `condition` is met, the intervention is initiated within `nperiod` time units. During the grace period, the treatment variable follows its natural value or initiate intervention with a uniform distribution at each time point.

The following is the user-defined intervention:

- User-defined Interventions: The output of the user-defined intervention should contain the intervened value for each individual at each time point, and should be of the same size as the number of rows in `data`.

Please see examples in the "Examples" section.

In order to obtain an inverse probability (IP) weighted natural course risk based on the observed data, users must specify a censoring variable through `censor_name` and a corresponding censoring model through `censor_model`. Please see Chiu et al. (2023) for more details regarding the IP weighted estimate of the natural course risk.

If competing event exists in the data, users need to specify the name of the competing variable through `competing_name` and the model specification through `competing_model` for hazard-based ICE estimator. Users need to specify whether to treat the competing event as censoring or total effect through `total_effect`.

We provide flexible term options in model specification for the outcome, censoring, competing, and hazard model. Users could specify polynomial terms using functions `I` and `poly` and spline terms using `ns` from splines package and `rcspline.eval` from Hmisc package. In addition, users could specify lagged terms using the format lagn_`var` to indicate lagging the variable *var* with *n* periods. If the lagged variable is a treatment variable, this variable is automatically intervened based on user-defined intervention. The polynomial and spline terms could be used on lagged variables.

Value

A list containing the following components. Each component that contains the fitted models includes the model fits, the summary of the fitted model, standard errors of the coefficients, variance-covariance matrices of the parameters, and the root mean square error (RMSE) values.

`estimator.type` A string describing the type of the estimator.

`summary` A summary table containing the estimated risk, risk ratio, and risk difference for user-defined interventions including estimated natural course risk and the observed risk. If `nsamples` is greater than 0, the summary table includes standard error and confidence interval for the point estimates.

`risk.over.time` A data frame containing the estimated risk at each time point for each intervention.

`initial.outcome` A list, where the name of each sublist corresponds to each specified intervention description, and each sublist contains the fitted models for the outcome model in the first step of algorithm.

`initial.comp` A list, where the name of each sublist corresponds to each specified intervention description, and each sublist contains the fitted models for the competing model in the first step of algorithm (if applicable).

`np.risk.model` A list containing the fitted models for the censoring and/or competing model in estimating observed risk (if applicable).

`outcome.models.by.step` A list, where the name of each sublist corresponds to each specified intervention description, and each sublist contains the fitted models for the outcome model in each iteration of algorithm.

`comp.models.by.step` A list, where the name of each sublist corresponds to each specified intervention description, and each sublist contains the fitted models for the competing model in each iteration of algorithm (if applicable).

hazard.models.by.step
: A list, where the name of each sublist corresponds to each specified intervention description, and each sublist contains the fitted models for the hazard model (if applicable), either time-specific models at all time points or one pooled-over-time global model.

boot.data
: A list of bootstrap samples. If nsamples is set to 0, a NULL value is returned.

boot.initial.outcome
: A list, where the name of each sublist corresponds to each specified intervention description, and each sublist contains the fitted models for the outcome model in the first step of algorithm on the bootstrap samples. If nsamples is set to 0, a NULL value is returned.

boot.initial.comp
: A list, where the name of each sublist corresponds to each specified intervention description, and each sublist contains the fitted models for the competing model in the first step of algorithm on the bootstrap samples (if applicable). If nsamples is set to 0, a NULL value is returned.

boot.np.risk.model
: A list containing the fitted models for the censoring and/or competing model in estimating observed risk on the bootstrap samples (if applicable). If nsamples is set to 0, a NULL value is returned.

boot.outcome.models.by.step
: A list, where the name of each sublist corresponds to each specified intervention description, and each sublist contains the fitted models for the outcome model in each iteration of algorithm on the bootstrap samples (if applicable). If nsamples is set to 0, a NULL value is returned.

boot.comp.models.by.step
: A list, where the name of each sublist corresponds to each specified intervention description, and each sublist contains the fitted models for the competing model in each iteration of algorithm on the bootstrap samples (if applicable). If nsamples is set to 0, a NULL value is returned.

boot.hazard.models.by.step
: A list, where the name of each sublist corresponds to each specified intervention description, and each sublist contains the fitted models for the hazard model (if applicable), either time-specific models at all time points or one pooled-over-time global model, on the bootstrap samples. If nsamples is set to 0, a NULL value is returned.

References

Wen L, Young JG, Robins JM, Hernán MA. Parametric g-formula implementations for causal survival analyses. Biometrics. 2021;77(2):740-753.

McGrath S, Lin V, Zhang Z, Petito LC, Logan RW, Hernán MA, and JG Young. gfoRmula: An R package for estimating the effects of sustained treatment strategies via the parametric g-formula. Patterns. 2020;1:100008.

Young JG, Herńan MA, Robins JM. Identification, estimation and approximation of risk under interventions that depend on the natural value of treatment using observational data. Epidemiologic Methods. 2014;3(1):1-19.

Young JG, Vatsa R, Murray EJ, Hernán MA. Interval-cohort designs and bias in the estimation of per-protocol effects: a simulation study. Trials. 2019;20(1):552.

Díaz, I, Williams, N, Hoffman, KL, & Schenck, EJ. Nonparametric causal effects based on longitudinal modified treatment policies. Journal of the American Statistical Association. 2021;118(542), 846–857.

Young JG, Stensrud MJ, Tchetgen Tchetgen EJ, Hernán MA. A causal framework for classical statistical estimands in failure-time settings with competing events. Statistics in medicine. 2020;39(8):1199-1236.

Wen L, Hernán MA, Robins JM. Multiply robust estimators of causal effects for survival outcomes. Scandinavian journal of statistics, theory and applications. 2022;49(3):1304-1328.

Haneuse S, Rotnitzky A. Estimation of the effect of interventions that modify the received treatment. Statistics in medicine. 2013;32(30):5260-5277.

McGrath S, Young JG, Hernán MA. Revisiting the g-null Paradox. Epidemiology. 2022;33(1):114-120.

Chiu YH, Wen L, McGrath S, Logan R, Dahabreh IJ, Hernán MA. Evaluating model specification when using the parametric g-formula in the presence of censoring. American Journal of Epidemiology. 2023;192:1887–1895.

Examples

```
data <- gfoRmulaICE::compData

# Example 1: Dynamic Intervention

# We consider the following interventions and intervened at all time points.
# Intervention 1 on A2: at time t, if L1 = 0, then treat; otherwise, not treat.
# Intervention 2 on A2: never treat upon until L1 = 0, after which follows always treat.
# Intervention 3 on A2: never treat upon until L1 = 0, after which follows natural course.

# We use classical pooled ICE estimator,
# natural course as the reference intervention, and the following models:
# a. outcome model: Y ~ L1 + L2 + A1 + A2
# b. censor model: C ~ L1 + L2 + A1 + A2
# c. competing model: D ~ L1 + L2 + A1 + A2.
# We estimate variance using bootstrap with 1000 replicates, normal quantile, and parallel computing.

ice_fit1 <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
ref_idx = 0,
estimator = pool(hazard = F),
nsamples = 1000, ci_method = "percentile",
parallel = T, ncores = 5,
int_descript = c("Dynamic Intervention 1", "Dynamic Intervention 2",
"Dynamic Intervention 3"),
intervention1.A2 = list(dynamic("L1 == 0", static(0), static(1))),
intervention2.A2 = list(dynamic("L1 == 0", static(0), static(1), absorb = T)),
intervention3.A2 = list(dynamic("L1 == 0", static(0), natural_course()))
)

plot_risk(ice_fit1)
```

```
# Example 2: Built-in Interventions

# We consider the following interventions and intervene at all time points.
# Intervention 1 on A1: always treat with value 3.
# Intervention 1 on A2: always treat with value 1.
# Intervention 2 on L2: when the natural value of L2 at time t is lower than -3, set its value to -3.
# Otherwise, do not intervene.
# Intervention 3 on A2: dynamic intervention (treat when L1 = 0) with uniform grace period of 2 periods

# We use classical pooled ICE estimator,
# natural course as the reference intervention, and the following models:
# a. outcome model: Y ~ L1 + L2 + A1 + A2
# b. censor model: C ~ L1 + L2 + A1 + A2
# c. competing model: D ~ L1 + L2 + A1 + A2.
# We estimate variance using bootstrap with 1000 replicates, normal quantile, and parallel computing.

ice_fit2 <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
ref_idx = 0,
estimator = pool(hazard = F),
nsamples = 1000, ci_method = "percentile",
parallel = T, ncores = 5,
int_descript = c("Static Intervention", "Threshold Intervention",
"Dynamic Intervention with Grace Period"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.L2 = list(threshold(-3, Inf)),
intervention3.A2 = list(grace_period("uniform", 2, "L1 == 0"))
)

plot_risk(ice_fit2)

# Example 3: User-defined Intervention

# We consider the following interventions and intervene at all time points.
# Intervention 1 on A1: always treat with value 3.
# Intervention 1 on A2: always treat with value 1.
# Intervention 2 on A1: at time t, if L2 < 0, then assign 1; if 0 <= L2 < 2, then assign 2; otherwise, assign 3.
# Intervention 2 on A2: at time t, if L1 = 0, then treat; otherwise, not treat.

# We use classical pooled ICE estimator,
# natural course as the reference intervention, and the following models:
# a. outcome model: Y ~ L1 + L2 + A1 + A2
# b. censor model: C ~ L1 + L2 + A1 + A2
# c. competing model: D ~ L1 + L2 + A1 + A2.
# We estimate variance using bootstrap with 1000 replicates and percentile quantile.

dynamic_cat <- case_when(data$L2 < 0 ~ 1,
data$L2 >= 0 & data$L2 < 2 ~ 2, T ~ 3)

ice_fit3 <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
```

```
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
ref_idx = 0,
estimator = pool(hazard = F),
nsamples = 1000, ci_method = "percentile",
parallel = T, ncores = 5,
int_descript = c("Static Intervention", "Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("L1 == 0", static(0), static(1)))
)

plot_risk(ice_fit3)

# Example 4: Different ICE Estimators

# We use the interventions in Example 3 and implement each ICE estimator.

# a. hazard-based pooled ICE:
# hazard model is time-specific and shares the same model statement as the outcome model

ice_fit4a <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
competing_model = D ~ L1 + L2 + A1 + A2,
ref_idx = 0,
estimator = pool(hazard = T),
nsamples = 1000, ci_method = "percentile",
parallel = T, ncores = 5,
int_descript = c("Static Intervention", "Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("L1 == 0", static(0), static(1)))
)

plot_risk(ice_fit4a)

# b. hazard-based pooled ICE:
# hazard model is time-specific and uses Y ~ L1 + L2

ice_fit4b <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
competing_model = D ~ L1 + L2 + A1 + A2,
```

```
hazard_model = Y ~ L1 + L2,
ref_idx = 0,
estimator = pool(hazard = T),
nsamples = 1000, ci_method = "percentile",
parallel = T, ncores = 5,
int_descript = c("Static Intervention", "Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("L1 == 0", static(0), static(1)))
)

plot_risk(ice_fit4b)

# c. hazard-based pooled ICE:
# hazard model is pooled-over-time and includes flexible terms of time variable

library(splines)

ice_fit4c <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
competing_model = D ~ L1 + L2 + A1 + A2,
hazard_model = Y ~ L1 + L2 + A1 + A2 + ns(t0, df = 2),
global_hazard = T,
ref_idx = 0,
estimator = pool(hazard = T),
nsamples = 1000, ci_method = "percentile",
parallel = T, ncores = 5,
int_descript = c("Static Intervention", "Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("L1 == 0", static(0), static(1)))
)

plot_risk(ice_fit4c)

# d. classical stratified ICE:

ice_fit4d <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2,
censor_model = C ~ L1 + L2,
ref_idx = 0,
estimator = strat(hazard = F),
nsamples = 1000, ci_method = "percentile",
parallel = T, ncores = 5,
int_descript = c("Static Intervention", "Dynamic Intervention"),
intervention1.A1 = list(static(3)),
```

```
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("L1 == 0", static(0), static(1)))
)

plot_risk(ice_fit4d)

# e. hazard-based stratified ICE:
# hazard model is time-specific and uses Y ~ L1
# (Note: a pooled-over-time hazard model is not valid for stratified ICE.)

ice_fit4e <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2,
censor_model = C ~ L1 + L2,
competing_model = D ~ L1 + L2,
hazard_model = Y ~ L1,
ref_idx = 0,
estimator = strat(hazard = T),
nsamples = 1000, ci_method = "percentile",
parallel = T, ncores = 5,
int_descript = c("Static Intervention: Model 1",
"Dynamic Intervention: Model 1"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("L1 == 0", static(0), static(1)))
)

plot_risk(ice_fit4e)


# f. doubly robust ICE:

ice_fit4f <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2,
censor_model = C ~ L1 + L2,
ref_idx = 0,
estimator = weight(list(A1 ~ L1 + L2, A2 ~ L1 + L2)),
nsamples = 1000, ci_method = "percentile",
parallel = T, ncores = 5,
int_descript = c("Static Intervention",
"Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("L1 == 0", static(0), static(1)))
)

plot_risk(ice_fit4f)
```

```
# g. hazard-based stratified ICE with intervention-specific models:
# hazard model is time-specific and same as the outcome model
# consider the total effect for competing event,
# using normal quantile for variance estimates,
# and the following outcome models and competing models:
# outcome model for intervention 1: Y ~ L1,
# outcome model for intervention 2: Y ~ L1 + L2,
# competing model for intervention 1: D ~ L1 + L2,
# competing model for intervention 2: D ~ L1

ice_fit4g <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
outcome_model = Y ~ L1, censor_model = C ~ L1,
competing_model = D ~ L1,
comp_effect = 1,
ref_idx = 0,
estimator = strat(hazard = T),
nsamples = 1000, ci_method = "normal",
parallel = T, ncores = 5,
int_descript = c("Static Intervention: Model 2",
"Dynamic Intervention: Model 2"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("L1 == 0", static(0), static(1))),
outcomeModel.1 = Y ~ L1 + L2,
compModel.2 = D ~ L1 + L2
)

# Compare with the ICE estimates in Example 4e:
plot_risk(ice_fit4e, ice_fit4g)
summary_table(ice_fit4e, ice_fit4g)

# Example 5: Flexible Model Specification

# a. Complicated terms in model statement:
# We use the same interventions and ICE estimator in Example 3,
# and include polynomial, spline, and lagged terms in models.

ice_fit5a <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ I(L1^2) + rcspline.eval(lag1_L2, knots = 1:3) + A1 + A2,
censor_model = C ~ lag1_L1 + poly(L2, degree = 2) + A1 + A2,
ref_idx = 0,
estimator = pool(hazard = F),
nsamples = 1000, ci_method = "percentile",
parallel = T, ncores = 5,
int_descript = c("Static Intervention", "Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
```

```
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("L1 == 0", static(0), static(1)))
)

plot_risk(ice_fit5a)

# b. Using static intervention as reference:
# We use the same interventions and ICE estimator in Example 3,
# but use static intervention as the reference intervention.

ice_fit5b <- ice(data = data, time_points = 4,
id = "id", time_name = "t0",
censor_name = "C", outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ I(L1^2) + rcspline.eval(lag1_L2, knots = 1:3) + A1 + A2,
censor_model = C ~ lag1_L1 + poly(L2, degree = 2) + A1 + A2,
ref_idx = 1,
estimator = pool(hazard = F),
nsamples = 1000, ci_method = "percentile",
parallel = T, ncores = 5,
int_descript = c("Static Intervention", "Dynamic Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1)),
intervention2.A1 = list(dynamic_cat),
intervention2.A2 = list(dynamic("L1 == 0", static(0), static(1)))
)

plot_risk(ice_fit5b)
```

---

natural_course *Natural Course*

Description

This function specifies the natural course intervention on the treatment variable in `data`.

Usage

```
natural_course(data = interv_data, treat_var = treatment_varname)
```

Arguments

`data`	a data frame containing the observed data.
`treat_var`	a string specifying the treatment variable in `data`.

Value

a vector containing the intervened values of the same size as the number of rows in `data`.

Examples

```
data <- gfoRmulaICE::compData
natural_course <- natural_course(data = data, treat_var = "A")
```

---

plot.ICE   *Plot method for ICE estimator objects*

Description

This function provides visualization of estimated risk for all specified interventions, estimated natural course risk, and observed risk at each time point.

Usage

```
## S3 method for class 'ICE'
plot(..., plot_obs = T, label = 0)
```

Arguments

`...`	ICE estimator objects.
`plot_obs`	a logical value indicating whether to plot the observed risk over time. Default is `TRUE`.
`label`	a number specifying which time label is used in x-axis. 0 represents using generic numerical time index, and 1 represents using the original time label from the data. Default is 0.

Value

a plot for risks of all the interventions specified in `...`.

Examples

```
data <- gfoRmulaICE::compData

ice_fit1 <- ice(
data = data,
time_points = 4,
id = "id",
time_name = "t0",
censor_name = "C",
outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
ref_idx = 0,
estimator = pool(hazard = F),
int_descript = "Static Intervention",
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1))
)
```

```
ice_fit2 <- ice(
data = data,
time_points = 4,
id = "id",
time_name = "t0",
censor_name = "C",
outcome_name = "Y",
compevent_name = "D",
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
competing_model = D ~ L1 + L2 + A1 + A2,
ref_idx = 0,
estimator = pool(hazard = T),
int_descript = "Static Intervention",
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1))
)

plot(ice_fit1, ice_fit2)
```

`pool` *Indicator for the pooling over treatment history ICE estimator*

Description

This function identifies the pooling over treatment history ICE estimator. The classical pooling over treatment history ICE estimator is specified by `pool(hazard = F)`. The hazard based pooling over treatment history ICE estimator is specified by `pool(hazard = T)`.

Usage

```
pool(hazard)
```

Arguments

`hazard`	a logical value indicating whether to use hazard-based ICE estimator.

Value

a logical value on whether to use hazard-based ICE estimator.

`static` *Static*

Description

This function specifies the static intervention, either treat with a constant value or never treat, on the treatment variable in `data`.

Usage

```
static(value, data = interv_data)
```

Arguments

`value`	a number specifying the intervention value. 0 for never treat.
`data`	a data frame containing the observed data.

Value

a vector containing the intervened values of the same size as the number of rows in `data`.

Examples

```
data <- gfoRmulaICE::compData
always_treat <- static(value = 1, data = data)
```

`strat` *Indicator for the stratifying on treatment history ICE estimator*

Description

This function identifies the stratifying on treatment history ICE estimator. The classical stratifying on treatment history ICE estimator is specified by `strat(hazard = F)`. The hazard based stratifying on treatment history ICE estimator is specified by `strat(hazard = T)`.

Usage

```
strat(hazard)
```

Arguments

`hazard`	a logical value indicating whether to use hazard-based ICE estimator.

Value

a logical value on whether to use hazard-based ICE estimator.

summary.ICE *Summary method for ICE Estimator Objects*

Description

This function returns a summary table for ICE estimator objects.

Usage

```
## S3 method for class 'ICE'
summary(...)
```

Arguments

...	the ICE estimator objects.

Value

a data frame containing the summary table for all specified ICE estimator objects.

Examples

```
data <- gfoRmulaICE::compData

fit_classical_pool <- ice(
data = data,
K = 4,
id = "id",
time_name = "t0",
outcome_name = "Y",
censor_name = "C",
competing_name = "D",
estimator = pool(hazard = F),
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
ref_idx = 0,
int_descript = c("Static Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1))
)

fit_hazard_pool <- ice(
data = data,
K = 4,
id = "id",
time_name = "t0",
outcome_name = "Y",
censor_name = "C",
competing_name = "D",
estimator = pool(hazard = T),
comp_effect = 0,
outcome_model = Y ~ L1 + L2 + A1 + A2,
censor_model = C ~ L1 + L2 + A1 + A2,
```

```
competing_model = D ~ L1 + L2 + A1 + A2,
ref_idx = 0,
int_descript = c("Static Intervention"),
intervention1.A1 = list(static(3)),
intervention1.A2 = list(static(1))
)

summary(fit_classical_pool, fit_hazard_pool)
```

`threshold` *Threshold*

Description

This function specifies the threshold intervention on the treatment variable in `data`. If treatment value is between the lower bound and the upper bound, it follows the natural value of the treatment. If treatment value is either below the lower bound or above the upper bound, it is set to the lower bound or the upper bound, correspondingly. See Young et al. (2014) for more details.

Usage

```
threshold(
  lower_bound,
  upper_bound,
  var = threshold_treatment,
  data = interv_data
)
```

Arguments

`lower_bound`	a number indicating the lower bound of the threshold.
`upper_bound`	a number indicating the upper bound of the threshold.
`var`	a string specifying the treatment variable for the intervention.
`data`	a data frame containing the observed data.

Value

a vector containing the intervened values of the same size as the number of rows in `data`.

References

Young JG, Herńan MA, Robins JM. Identification, estimation and approximation of risk under interventions that depend on the natural value of treatment using observational data. Epidemiologic Methods. 2014;3(1):1-19.

Examples

```
data <- gfoRmulaICE::compData
threshold_treat <- threshold(lower_bound = 0, upper_bound = 2, var = "A", data = data)
```

`weight` *Indicator for the doubly robust ICE estimator*

Description

This function identifies the doubly robust ICE estimator. The treatment models could be specified by `treat_model`.

Usage

```
weight(treat_model = list())
```

Arguments

`treat_model`	a list of formulas specifying the treatment model for the corresponding treatment variable. The length of list must match the number of treatment variables.

Value

treatment model specifications and treatment variable names.

Index